TU Clausthal

Clausthal University of Technology

Revised version. Revision date: 12.12.2008

Matrix adaptation in discriminative vector quantization

Petra Schneider[1], **Michael Biehl**[1], **Barbara Hammer**[2]

IfI Technical Report Series **IFI-08-08**

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Wojciech Jamroga
Contact: wjamroga@in.tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Matrix adaptation in discriminative vector quantization

Petra Schneider[1], Michael Biehl[1], Barbara Hammer[2]

[1]Institute for Mathematics and Computing Science, University of Groningen
P.O. Box 407, 9700 AK Groningen, The Netherlands
{p.schneider,m.biehl}@rug.nl

[2]Institute of Computing Science, Clausthal University of Technology
Julius Albert Strasse 4, 38678 Clausthal - Zellerfeld, Germany
hammer@in.tu-clausthal.de

Abstract

Discriminative vector quantization schemes such as learning vector quantization (LVQ) and extensions thereof offer efficient and intuitive classifiers which are based on the representation of classes by prototypes. The original methods, however, rely on the Euclidean distance corresponding to the assumption that the data can be represented by isotropic clusters. For this reason, extensions of the methods to more general metric structures have been proposed such as relevance adaptation in generalized LVQ (GLVQ) and matrix learning in GLVQ. In these approaches, metric parameters are learned based on the given classification task such that a data driven distance measure is found. In this article, we consider full matrix adaptation in advanced LVQ schemes; in particular, we introduce matrix learning to a recent statistical formalization of LVQ, robust soft LVQ, and we compare the results on several artificial and real life data sets to matrix learning in GLVQ, which constitutes a derivation of LVQ-like learning based on a (heuristic) cost function. In all cases, matrix adaptation allows a significant improvement of the classification accuracy. Interestingly, however, the principled behavior of the models with respect to prototype locations and extracted matrix dimensions shows several characteristic differences depending on the data sets.

Keywords: learning vector quantization, generalized LVQ, robust soft LVQ, metric adaptation

1 Introduction

Discriminative vector quantization schemes such as learning vector quantization (LVQ) constitute very popular classification methods due to their intuitivity and robustness: they represent the classification by (usually few) prototypes which constitute typical representatives of the respective classes and, thus, allow a direct inspection of the given classifier. Training often takes place by Hebbian learning such that very fast and simple

training algorithms result. Further, unlike the perceptron or the support vector machine, LVQ provides an integrated and intuitive classification model for any given number of classes. Numerous modifications of original LVQ exist which extend the basic learning scheme as proposed by Kohonen towards adaptive learning rates, faster convergence, or better approximation of Bayes optimal classification, to name just a few [8]. Despite their popularity and efficiency, most LVQ schemes are solely based on heuristics and a deeper mathematical investigation of the models has just recently been initiated. On the one hand, their worst case generalization capability can be limited in terms of general margin bounds using techniques from computational learning theory [5, 4]. On the other hand, the characteristic behavior and learning curves of popular models can exactly be investigated in typical model situations using the theory of online learning [2]. Many questions, however, remain unsolved such as convergence and typical prototype locations of heuristic LVQ schemes in concrete, finite training settings.

Against this background, researchers have proposed variants of LVQ which can directly be derived from an underlying cost function which is optimized during training e.g. by means of a stochastic gradient ascent/descent. Generalized LVQ (GLVQ) as proposed by Sato and Yamada constitutes one example [10]: its intuitive (though heuristic) cost function can be related to a minimization of classification errors and, at the same time, a maximization of the hypothesis margin of the classifier which characterizes its generalization ability [5]. The resulting algorithm is indeed very robust and powerful, however, an exact mathematical analysis is still lacking. A very elegant and mathematically well-founded alternative has been proposed by Seo and Obermayer: in [12], a statistical approach is introduced which models given classes as mixtures of Gaussians. Prototype parameters are optimized by maximizing the likelihood ratio of correct versus incorrect classification. A learning scheme which closely resembles LVQ2.1 results. This cost function, however, is unbounded such that numerical instabilities occur which, in practice, cause the necessity of restricting updates to data from a *window* close to the decision boundary. The approach of [12] offers an elegant alternative: A robust optimization scheme is derived from a maximization of the likelihood ratio of the probability of correct classification versus the *total* probability in a Gaussian mixture model. The resulting learning scheme, robust soft LVQ (RSLVQ), leads to an alternative discrete LVQ scheme where prototypes are adapted solely based on misclassifications.

RSLVQ constitutes a very attractive model due to the fact that all underlying model assumptions are stated explicitly in the statistical formulation – and, they can easily be changed if required by the application scenario. Besides, the resulting model shows superior classification accuracy compared to GLVQ in a variety of settings as we will demonstrate in this article.

All these methods, however, suffer from the problem that classification is based on a predefined metric. The use of Euclidean distance, for instance, corresponds to the implicit assumption of isotropic clusters. Such models can only be successful if the data displays a Euclidean characteristic. This is particularly problematic for high-dimensional data where noise accumulates and disrupts the classification, or heterogeneous data sets where different scaling and correlations of the dimensions can be

observed. Thus, a more general metric structure would be beneficial in such cases. The field of metric adaptation constitutes a very active research topic in various distance based approaches such as unsupervised or semi-supervised clustering and visualization [1, 7], k-nearest neighbor approaches [14, 15] and learning vector quantization [6, 11]. We will focus on matrix learning in LVQ schemes which accounts for pairwise correlations of features, i.e. a very general and flexible set of classifiers. On the one hand, we will investigate the behavior of generalized matrix LVQ (GMLVQ) in detail, a matrix adaptation scheme for GLVQ which is based on a heuristic, though intuitive cost function. On the other hand, we will develop matrix adaptation for RSLVQ, a statistical model for LVQ schemes, and thus we will arrive at a uniform statistical formulation for prototype and metric adaptation in discriminative prototype-based classifiers. We will introduce variants which adapt the matrix parameters globally based on the training set or locally for every given prototype or mixture component, respectively.

Matrix learning in RSLVQ and GLVQ will be evaluated and compared in a variety of learning scenarios: first, we consider test scenarios where prior knowledge about the form of the data is available. Furthermore, we compare the methods on several benchmarks from the UCI repository [9].

Interestingly, depending on the data, the methods show different characteristic behavior with respect to prototype locations and learned metrics. Although the classification accuracy is in many cases comparable, they display quite different behavior concerning their robustness with respect to parameter choices and the characteristics of the solutions. We will point out that these findings have consequences on the interpretability of the results. In all cases, however, matrix adaptation leads to an improvement of the classification accuracy, despite a largely increased number of free parameters.

2 Advanced learning vector quantization schemes

Learning vector quantization has been introduced by Kohonen [8], and a variety of extensions and generalizations exist. Here we focus on approaches based on a cost function, i.e. generalized learning vector quantization (GLVQ) and robust soft learning vector quantization (RSLVQ).

Assume training data $\{\boldsymbol{\xi}_i, y_i\}_{i=1}^{l} \in \mathbb{R}^N \times \{1, \ldots, C\}$ are given, N denoting the data dimensionality and C the number of different classes. An LVQ network $W = \{(\boldsymbol{w}_j, c(\boldsymbol{w}_j)) : \mathbb{R}^N \times \{1, \ldots, C\}\}_{j=1}^{m}$ consists of a number m of prototypes $\boldsymbol{w} \in \mathbb{R}^N$ which are characterized by their location in feature space and their class label $c(\boldsymbol{w}) \in \{1 \ldots, C\}$. Classification is based on a winner takes all scheme. A data point $\boldsymbol{\xi} \in \mathbb{R}^N$ is mapped to the label $c(\boldsymbol{\xi}) = c(\boldsymbol{w}_i)$ of the prototype, for which $d(\boldsymbol{\xi}, \boldsymbol{w}_i) \leq d(\boldsymbol{\xi}, \boldsymbol{w}_j)$ holds $\forall j \neq i$. Here d is an appropriate distance measure. Hence, $\boldsymbol{\xi}$ is mapped to the class of the closest prototype, the so-called winner. Often, d is chosen as the squared Euclidean metric, i.e. $d(\boldsymbol{\xi}, \boldsymbol{w}) = (\boldsymbol{\xi} - \boldsymbol{w})^T(\boldsymbol{\xi} - \boldsymbol{w})$.

LVQ algorithms aim at an adaptation of the prototypes such that a given data set is classified as accurately as possible. The first LVQ schemes proposed heuristic adaptation rules based on the principle of Hebbian learning, such as *LVQ2.1*, which, for

a given data point $\boldsymbol{\xi}$, adapts the closest prototype $\boldsymbol{w}^+(\boldsymbol{\xi})$ with the same class label $c(\boldsymbol{w}^+(\boldsymbol{\xi})) = c(\boldsymbol{\xi})$ into the direction of $\boldsymbol{\xi}$: $\Delta\boldsymbol{w}^+(\boldsymbol{\xi}) = \alpha \cdot (\boldsymbol{\xi} - \boldsymbol{w}^+(\boldsymbol{\xi}))$ and the closest incorrect prototype $\boldsymbol{w}^-(\boldsymbol{\xi})$ with a different class label $c(\boldsymbol{w}^-(\boldsymbol{\xi})) \neq c(\boldsymbol{\xi})$ is moved into the opposite direction: $\Delta\boldsymbol{w}^- = -\alpha \cdot (\boldsymbol{\xi} - \boldsymbol{w}^-(\boldsymbol{\xi}))$. Here, $\alpha > 0$ is the learning rate. Since, often, LVQ2.1 shows divergent behavior, a window rule is introduced, and adaptation takes place only if $\boldsymbol{w}^+(\boldsymbol{\xi})$ and $\boldsymbol{w}^-(\boldsymbol{\xi})$ are the closest two prototypes of $\boldsymbol{\xi}$.

Generalized LVQ derives a similar update rule from the following cost function:

$$E_{\mathrm{GLVQ}} = \sum_{i=1}^{l} \Phi(\mu(\boldsymbol{\xi}_i)) = \sum_{i=1}^{l} \Phi\left(\frac{d(\boldsymbol{\xi}_i, \boldsymbol{w}^+(\boldsymbol{\xi}_i)) - d(\boldsymbol{\xi}_i, \boldsymbol{w}^-(\boldsymbol{\xi}_i))}{d(\boldsymbol{\xi}_i, \boldsymbol{w}^+(\boldsymbol{\xi}_i)) + d(\boldsymbol{\xi}_i, \boldsymbol{w}^-(\boldsymbol{\xi}_i))}\right). \tag{1}$$

Φ is a monotonic function such as the logistic function or the identity which is used throughout the following. The numerator of a single summand is negative if the classification of $\boldsymbol{\xi}$ is correct. Further, a small value corresponds to a classification with large margin, i.e. large difference of the distance to the closest correct and incorrect prototype. In this sense, GLVQ tries to minimize the number of misclassifications and to maximize the margin of the classification. The denominator accounts for a scaling of the terms such that the arguments of Φ are restricted to the interval $(-1, 1)$ and numerical problems are avoided. The cost function of GLVQ can be related to a compromise of the minimization of the training error and the generalization ability of the classifier which is determined by the hypothesis margin (see [4, 5]). The connection, however, is not exact. The update formulas of GLVQ can be derived by means of the gradients of E_{GLVQ} (see [10]). Interestingly, the learning rule resembles LVQ2.1 in the sense that the closest correct prototype is moved towards the considered data point and the closest incorrect prototype is moved away from the data point. The size of this adaptation step is determined by the magnitude of terms stemming from E_{GLVQ}; this change accounts for a better robustness of the algorithm compared to LVQ2.1.

Unlike GLVQ, *robust soft learning vector quantization* is based on a statistical modelling of the situation which makes all assumptions explicit: the probability density of the underlying data distribution is described by a mixture model. Every component j of the mixture is assumed to generate data which belongs to only one of the C classes. The probability density of the full data set is given by

$$p(\boldsymbol{\xi}|W) = \sum_{i=1}^{C} \sum_{j:c(\boldsymbol{w}_j)=i}^{m} p(\boldsymbol{\xi}|j)P(j), \tag{2}$$

where the conditional density $p(\boldsymbol{\xi}|j)$ is a function of prototype $\boldsymbol{w}_j$. For example, the conditional density can be chosen to have the normalized exponential form $p(\boldsymbol{\xi}|j) = K(j) \cdot \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)$, and the prior $P(j)$ can be chosen identical for every prototype $\boldsymbol{w}_j$. RSLVQ aims at a maximization of the likelihood ratio:

$$E_{\mathrm{RSLVQ}} = \sum_{i=1}^{l} \log\left(\frac{p(\boldsymbol{\xi}_i, y_i|W)}{p(\boldsymbol{\xi}_i|W)}\right), \tag{3}$$

where $p(\boldsymbol{\xi}_i, y_i|W)$ is the probability density that $\boldsymbol{\xi}_i$ is generated by a mixture component of the correct class y_i and $p(\boldsymbol{\xi}_i|W)$ is the total probability density of $\boldsymbol{\xi}_i$. This implies

$$p(\boldsymbol{\xi}_i, y_i|W) = \sum_{j:c(\boldsymbol{w}_j)=y_i} p(\boldsymbol{\xi}_i|j)\, P(j), \quad p(\boldsymbol{\xi}_i|W) = \sum_j p(\boldsymbol{\xi}_i|j)\, P(j). \tag{4}$$

The learning rule of RSLVQ is derived from E_{RSLVQ} by a stochastic gradient ascent. Since the value of E_{RSLVQ} depends on the position of all prototypes, the complete set of prototypes is updated in each learning step. The gradient of a summand of E_{RSLVQ} for data point $(\boldsymbol{\xi}, y)$ with respect to a prototype $\boldsymbol{w}_j$ is given by (see the appendix)

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{w}_j}\left(\log \frac{p(\boldsymbol{\xi}, y|W)}{p(\boldsymbol{\xi}|W)}\right) = {} & \delta_{y,c(\boldsymbol{w}_j)}\,(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))\,\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)}{\partial \boldsymbol{w}_j} \\ & - (1 - \delta_{y,c(\boldsymbol{w}_j)})\, P(j|\boldsymbol{\xi})\,\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)}{\partial \boldsymbol{w}_j},\end{aligned} \tag{5}$$

where the Kronecker symbol $\delta_{y,c(\boldsymbol{w}_j)}$ tests whether the labels y and $c(\boldsymbol{w}_j)$ coincide. In the special case of a Gaussian mixture model with $\sigma_j^2 = \sigma^2$ and $P(j) = 1/m$ for all j, we obtain

$$f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2) = \frac{-d(\boldsymbol{\xi}, \boldsymbol{w})}{2\sigma^2}, \tag{6}$$

where $d(\boldsymbol{\xi}, \boldsymbol{w})$ is the distance measure between data point $\boldsymbol{\xi}$ and prototype $\boldsymbol{w}$. Original RSLVQ is based on the squared Euclidean distance. This implies

$$f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2) = -\frac{(\boldsymbol{\xi} - \boldsymbol{w})^T(\boldsymbol{\xi} - \boldsymbol{w})}{2\sigma^2}, \quad \frac{\partial f}{\partial \boldsymbol{w}} = \frac{1}{\sigma^2}(\boldsymbol{\xi} - \boldsymbol{w}). \tag{7}$$

Substituting the derivative of f in equation (5) yields the update rule for the prototypes in RSLVQ

$$\Delta \boldsymbol{w}_j = \frac{\alpha_1}{\sigma^2}\begin{cases} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi})(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) \neq y, \end{cases} \tag{8}$$

where $\alpha_1 > 0$ is the learning rate. In the limit of vanishing softness σ^2, the learning rule reduces to an intuitive crisp *learning from mistakes* (LFM) scheme, as pointed out in [12]: in case of erroneous classification, the closest correct and the closest wrong prototype are adapted along the direction pointing to / from the considered data point. Thus, a learning scheme very similar to LVQ2.1 results which reduces adaptation to wrongly classified inputs close to the decision boundary. While the soft version as introduced in [12] leads to a good classification accuracy as we will see in experiments, the limit rule has some principled deficiencies as shown in [2].

3 Matrix learning in advanced LVQ schemes

The squared Euclidean distance gives rise to isotropic clusters, hence the metric is not appropriate if data dimensions show a different scaling or correlations. A more general form can be obtained by extending the metric by a full matrix

$$d_\Lambda(\boldsymbol{\xi}, \boldsymbol{w}) = (\boldsymbol{\xi} - \boldsymbol{w})^T \Lambda (\boldsymbol{\xi} - \boldsymbol{w}), \tag{9}$$

where Λ is an $N \times N$-matrix which is restricted to positive definite forms to guarantee metricity. We can achieve this by substituting $\Lambda = \Omega^T \Omega$, where $\Omega \in \mathbb{R}^{M \times N}$. Further, Λ has to be normalized after each learning step to prevent the algorithm from degeneration. Two possible approaches are to restrict $\sum_i \Lambda_{ii}$ or $\det(\Lambda)$ to a fixed value, i.e. either the sum of eigenvalues or the product of eigenvalues is constant. Note that normalizing $\det(\Lambda)$ requires $M \geq N$, since otherwise Λ would be singular. In this work, we always set $M = N$. Since an optimal matrix is not known beforehand for a given classification task, we adapt Λ or Ω, respectively, during training. For this purpose, we substitute the distance in the cost functions of LVQ by the new measure

$$d_\Lambda(\boldsymbol{\xi}, \boldsymbol{w}) = \sum_{i,j,k} (\xi_i - w_i) \Omega_{ki} \Omega_{kj} (\xi_j - w_j). \tag{10}$$

Generalized matrix LVQ (GMLVQ) extends the cost function E_{GLVQ} by this more general metric and adapts the matrix parameters Ω_{ij} together with the prototypes by means of a stochastic gradient descent. The derivatives

$$\frac{\partial d_\Lambda(\boldsymbol{\xi}, \boldsymbol{w})}{\partial \boldsymbol{w}} = -2\Lambda(\boldsymbol{\xi} - \boldsymbol{w}), \qquad \frac{\partial d_\Lambda(\boldsymbol{\xi}, \boldsymbol{w})}{\partial \Omega_{lm}} = 2 \sum_i (\xi_i - w_i) \Omega_{li} (\xi_m - w_m)$$

yield the GMLVQ update rules

$$\Delta \boldsymbol{w}^+ = \alpha_1 \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \mu^+(\boldsymbol{\xi}) \cdot \Lambda \cdot d_\Lambda(\boldsymbol{\xi}, \boldsymbol{w}^+(\boldsymbol{\xi})), \tag{11}$$

$$\Delta \boldsymbol{w}^- = -\alpha_1 \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \mu^-(\boldsymbol{\xi}) \cdot \Lambda \cdot d_\Lambda(\boldsymbol{\xi}, \boldsymbol{w}^-(\boldsymbol{\xi})), \tag{12}$$

$$\begin{aligned} \Delta \Omega_{lm} = -\alpha_2 \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \\ \Big(\mu^+(\boldsymbol{\xi}) \cdot \Big([\Omega(\boldsymbol{\xi} - \boldsymbol{w}^+(\boldsymbol{\xi}))]_l \, (\xi_m - w_m^+) \Big) \\ - \mu^-(\boldsymbol{\xi}) \cdot \Big([\Omega(\boldsymbol{\xi} - \boldsymbol{w}^-(\boldsymbol{\xi}))]_l \, (\xi_m - w_m^-) \Big) \Big), \end{aligned} \tag{13}$$

where $\alpha_2 > 0$ is the learning rate for the metric parameter and $\mu^+(\boldsymbol{\xi}) = d(\boldsymbol{\xi}, \boldsymbol{w}^-(\boldsymbol{\xi}))/(d(\boldsymbol{\xi}, \boldsymbol{w}^+(\boldsymbol{\xi})) + d(\boldsymbol{\xi}, \boldsymbol{w}^-(\boldsymbol{\xi})))^2$, $\mu^-(\boldsymbol{\xi}) = d(\boldsymbol{\xi}, \boldsymbol{w}^+(\boldsymbol{\xi}))/(d(\boldsymbol{\xi}, \boldsymbol{w}^-(\boldsymbol{\xi})) + d(\boldsymbol{\xi}, \boldsymbol{w}^-(\boldsymbol{\xi})))^2$.

It is possible to introduce one global matrix Ω, which corresponds to a global transformation of the data space, or, alternatively, to introduce an individual matrix Ω_j for

every prototype. The latter corresponds to the possibility to adapt individual ellipsoidal clusters around every prototype. In this case, the squared distance is computed by

$$d(\boldsymbol{\xi}, \boldsymbol{w}_j) = (\boldsymbol{\xi} - \boldsymbol{w}_j)^T \Lambda_j (\boldsymbol{\xi} - \boldsymbol{w}_j). \tag{14}$$

Using this approach, the updates for the prototypes (Eq.s 12,13) include the local matrices $\Lambda^+,\ \Lambda^-$. For the metric parameters, the learning rules constitute

$$\begin{aligned}\Delta\Omega^+_{lm} &= -\alpha_2 \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \mu^+(\boldsymbol{\xi}) \cdot \\ &\qquad \Big([\Omega^+(\boldsymbol{\xi} - \boldsymbol{w}^+(\boldsymbol{\xi}))]_l \, (\xi_m - w^+_m)\Big), \qquad (15)\\ \Delta\Omega^-_{lm} &= \alpha_2 \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \mu^-(\boldsymbol{\xi}) \cdot \\ &\qquad \Big([\Omega^-(\boldsymbol{\xi} - \boldsymbol{w}^-(\boldsymbol{\xi}))]_l \, (\xi_m - w^-_m)\Big). \qquad (16)\end{aligned}$$

We refer to the extension of GMLVQ with local relevance matrices by the term *local GMLVQ* (LGMLVQ).
Now, we extend RSLVQ by the more general metric introduced in equation (9). The conditional density function obtains the form $p(\boldsymbol{\xi}|j) = K(j) \cdot \exp f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2, \Omega)$ with

$$f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2, \Omega) = \frac{-(\boldsymbol{\xi} - \boldsymbol{w})^T \Omega^T \Omega (\boldsymbol{\xi} - \boldsymbol{w})}{2\sigma^2}, \tag{17}$$

$$\frac{\partial f}{\partial \boldsymbol{w}} = \frac{1}{\sigma^2} \Omega^T \Omega \, (\boldsymbol{\xi} - \boldsymbol{w}) = \frac{1}{\sigma^2} \Lambda \, (\boldsymbol{\xi} - \boldsymbol{w}), \tag{18}$$

$$\frac{\partial f}{\partial \Omega_{lm}} = -\frac{1}{\sigma^2} \left(\sum_i (\xi_i - w_i) \Omega_{li} (\xi_m - w_m) \right). \tag{19}$$

Combining equations (5) and (18) yields the new update rule for the prototypes:

$$\Delta \boldsymbol{w}_j = \frac{\alpha_1}{\sigma^2} \begin{cases} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})) \, \Lambda \, (\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi}) \, \Lambda \, (\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) \neq y. \end{cases} \tag{20}$$

Taking the derivative of the summand E_{RSLVQ} for training sample $(\boldsymbol{\xi}, y)$ with respect to a matrix element Ω_{lm} leads us to the update rule (see the appendix)

$$\begin{aligned}\Delta\Omega_{lm} = &- \frac{\alpha_2}{\sigma^2} \cdot \\ &\sum_j \Bigg[\Big(\delta_{y,c(\boldsymbol{w}_j)} \left(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\right) - \left(1 - \delta_{y,c(\boldsymbol{w}_j)}\right) P(j|\boldsymbol{\xi}) \Big) \cdot \\ &\qquad \Big([\Omega(\boldsymbol{\xi} - \boldsymbol{w}_j)]_l \, (\xi_m - w_{j,m}) \Big) \Bigg], \qquad (21)\end{aligned}$$

where $\alpha_2 > 0$ is the learning rate for the metric parameters. The algorithm based on the update rules in equations (20) and (21) will be called *matrix RSLVQ* (MRSLVQ) in

the following. Similar to local matrix learning in GMLVQ, it is also possible to train an individual matrix Λ_j for every prototype. With individual matrices attached to all prototypes, the modification of (20) which includes the local matrices Λ_j is accompanied by (see the appendix)

$$\begin{aligned}\Delta\Omega_{j,lm} = -\frac{\alpha_2}{\sigma^2} \cdot \\ \Bigg[\Big(\delta_{y,c(\boldsymbol{w}_j)}(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})) - (1-\delta_{y,c(\boldsymbol{w}_j)})P(j|\boldsymbol{\xi})\Big)\cdot \\ \Big([\Omega_j(\boldsymbol{\xi}-\boldsymbol{w}_j)]_l\,(\xi_m - w_{j,m})\Big)\Bigg],\end{aligned} \tag{22}$$

under the constraint $K(j) = const.$ for all j. We term this learning rule *local MRSLVQ* (LMRSLVQ). Due to the restriction to constant normalization factors $K(j)$, the normalization $\det(\Lambda_j) = const.$ is assumed for this algorithm. Note that under the assumption of equal priors $P(j)$, a classifier using one prototype per class is still given by the standard LVQ classifier: $\boldsymbol{\xi} \mapsto c(\boldsymbol{w}_j)$ for which $d_{\Lambda_j}(\boldsymbol{\xi}, \boldsymbol{w}_j)$ is minimum. In more general settings, nearest prototype classification should coincide with the class of maximum likelihood ratio for most inputs since prototypes are usually distant from each other compared to the bandwidth σ^2. Interestingly, the generalization ability of this function class has been investigated in [11] including the possibility of adaptive local matrices. Worst case generalization bounds which depend on the number of prototypes and the hypothesis margin, i.e. the minimum difference between the closest correct and wrong prototype, can be found which are independent of the input dimensionality (in particular independent of the matrix dimensionality), such that good generalization capability can be expected from these classifiers. We will investigate this claim in several experiments. In addition, we will have a look at the robustness of the methods with respect to hyperparameters, the interpretability of the results, and the uniqueness of the learned matrices.

Although GLVQ and RSLVQ constitute two of the most promising theoretical derivations of LVQ schemes from global cost functions, they have so far not been compared in experiments. Further, matrix learning offers a striking extension of RSLVQ since it extends the underlying Gaussian mixture model towards the general form of arbitrary covariance matrices, which has not been introduced or tested so far. Thus, we are interested in several aspects and questions which should be highlighted by the experiments:

- What is the performance of the methods on real life data sets of different characteristics? Can the theoretically motivated claim of good generalization ability be substantiated by experiments?
- What is the robustness of the methods with respect to metaparameters such as σ^2?
- Do the methods provide meaningful (representative) prototypes or does the pro-

totype location change due to the specific learning rule in a *discriminative* approach?

- Are the extracted matrices meaningful? In how far do they differ between the approaches?
- Do there exist systematic differences in the solutions found by RSLVQ and GLVQ (with / without matrix adaptation)?

We first test the methods on two artificial data sets where the underlying density is known exactly, which are designed for the evaluation of matrix adaptation. Afterwards, we compare the algorithms on benchmarks from UCI [9].

4 Experiments

With respect to parameter initialization and learning rate annealing, we use the same strategies in all experiments. The mean values of random subsets of training samples selected from each class are chosen as initial states of the prototypes. The hyperparameter σ^2 is held constant in all experiments with RSLVQ, MRSLVQ and Local MRSLVQ. The learning rates are continuously reduced in the course of learning. We implement a schedule of the form

$$\alpha_i(t) = \frac{\alpha_i}{1 + c\,(t-1)} \tag{23}$$

($i \in \{1, 2\}$), where t counts the number training epochs. The factor c determines the speed of annealing and is chosen individually for every application. Special attention has to be paid to the normalization of the relevance matrices. With respect to the interpretability, it is advantageous to fix the sum of eigenvalues to a certain value. Besides, we observe that this approach shows a better performance and learning behaviour compared to the restriction of the matrices' determinant. For this reason, the normalization $\sum_i \Lambda_{ii} = 1$ is used for the applications in Sec. 4.1, since we do not discuss the adaptation of local matrices there. We initially set $\Lambda = \frac{1}{N} \cdot \mathbb{1}$, which results in d_Λ being equivalent to the Euclidean distance. Note that, in consequence, the Euclidean distance in RSLVQ and GLVQ has to be normalized to one as well to allow for a fair comparison with respect to learning rates. Accordingly, the RSLVQ- and GLVQ prototype updates and the function f in equation (7) have to be weighted by $1/N$.Training of local MRSLVQ in Sec. 4.2 requires the normalization $\det(\Lambda_j) = 1$. The local matrices Λ_j are initialized by the identity matrix in this case.

4.1 Artificial Data

In the first experiments, the algorithms are applied to the artificial data from [3] to illustrate the training of an LVQ-classifier based on the alternative cost functions with fixed and adaptive distance measure. The data sets 1 and 2 comprise three-class classification problems in a two dimensional space. Each class is split into two clusters with small or

large overlap, respectively (see Figure 1). We randomly select 2/3 of the data samples of each class for training and use the remaining data for testing. According to the a priori known distributions, the data is represented by two prototypes per class. Since we observe that the algorithms based on the RSLVQ cost function are very sensitive with respect to the learning parameter settings, slightly smaller values are chosen to train a classifier with (M)RSLVQ compared to G(M)LVQ. We use the settings

G(M)LVQ: $\alpha_1 = 5 \cdot 10^{-3},\ \alpha_2 = 1 \cdot 10^{-3},\ c = 1 \cdot 10^{-3}$
(M)RSLVQ: $\alpha_1 = 5 \cdot 10^{-4},\ \alpha_2 = 1 \cdot 10^{-4},\ c = 1 \cdot 10^{-3}$

and perform 1000 sweeps through the training set. The results presented in the following are averaged over 10 independent constellations of training and test set. We apply several different values σ^2 from the interval [0.001, 0.015] and present the simulations giving rise to the best mean performance on the training sets.

The results are summarized in Table 1. They are obtained with the hyperparameters settings $\sigma^2_{opt}(\text{RSLVQ}) = 0.002$ and $\sigma^2_{opt}(\text{MRSLVQ}) = 0.002, 0.003$ for data set 1 and 2, respectively. The use of the advanced distance measure yields only a slight improvement compared to the fixed Euclidean distance, since the distributions do not have favorable directions to classify the data. On data set 1, GLVQ and RSLVQ show nearly the same performance. However, the prototype configurations identified by the two algorithms vary significantly (see Figure 1). During GLVQ-training, the prototypes move close to the cluster centers in only a few training epochs, resulting in an appropriate approximation of the data by the prototypes. On the contrary, prototypes are frequently located outside the clusters, when the classifier is trained with the RSLVQ-algorithm. This behavior is due to the fact that only data points lying close to the decision boundary change the prototype constellation in RSLVQ significantly (see equation (8)). As depicted in Figure 2, only a small number of training samples are lying in the active region of the prototypes while the great majority of training samples attains only tiny weight values in equation (8) which are not sufficent to adjust the prototypes to the data in reasonable training time. This effect does not have negative impact on the classifica-

Table 1: Mean rate of misclassification (in %) obtained by the different algorithms on the artificial data sets 1 and 2 at the end of training. The values in brackets constitute the variances.

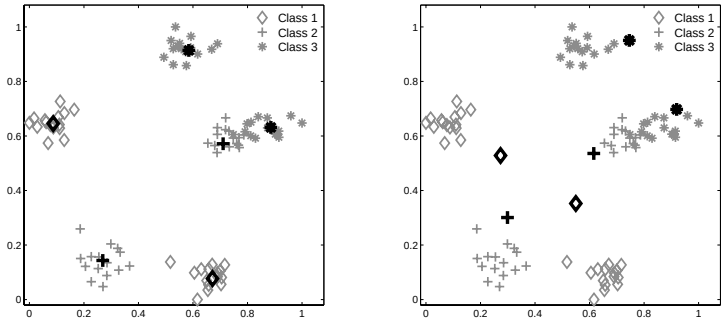
Algorithm	Data set 1		Data set 2	
	ε_{train}	ε_{test}	ε_{train}	ε_{test}
GLVQ	2.0 (0.02)	2.7 (0.07)	19.2 (0.9)	24.2 (1.9)
GMLVQ	2.0 (0.02)	2.7 (0.07)	18.6 (0.7)	23.0 (1.6)
RSLVQ	1.5 (0.01)	3.7 (0.04)	12.8 (0.07)	19.3 (0.3)
MRSLVQ	1.5 (0.01)	3.7 (0.02)	12.3 (0.04)	19.3 (0.3)

tion of the data set. However, the prototypes do not provide a reasonable approximation of the data.

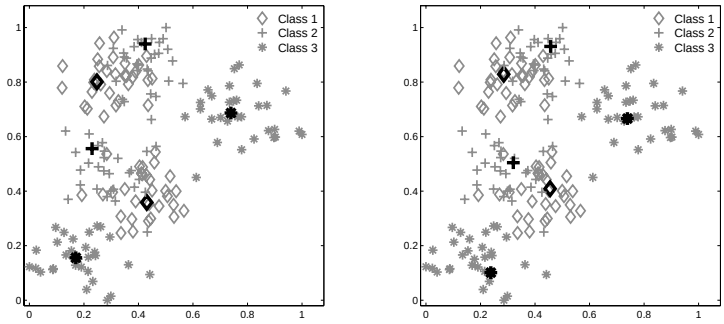
The prototype constellation identified by RSLVQ on data set 2 represents the classes clearly better (see Figure 1). Since the clusters show significant overlap, a sufficiently large number of training samples contributes to the learning process (see Figure 2) and the prototypes quickly adapt to the data. The good approximation of the data is accompanied by an improved classification performance compared to GLVQ. Although GLVQ also places prototypes close to the cluster centers, the use of the RSLVQ-cost function gives rise to the superior classifier for this data set. This observation is also confirmed by the experiments with GMLVQ and MRSLVQ.

To demonstrate the influence of metric learning, data set 3 is generated by embedding each sample $\boldsymbol{\xi} = (\xi_1, \xi_2) \in \mathbb{R}^2$ of data set 2 in $\mathbb{R}^{10}$ by choosing: $\xi_3 = \xi_1 + \eta_1, \ldots \xi_6 = \xi_1 + \eta_4$, where η_i comprises Gaussian noise with variances 0.05, 0.1, 0.2 and 0.5, respectively. The features $\xi_7, \ldots, \xi_{10}$ contain pure uniformly distributed noise in [-0.5, 0.5] and [-0.2, 0.2] and Gaussian noise with variances 0.5 and 0.2, respectively. Hence, the first two dimensions are most informative to classify this data set. The dimensions 3 to 6 still partially represent dimension 1 with increasing noise added. Finally, we apply a random linear transformation on the samples of data set 3 in order to construct a test scenario, where the discriminating structure is not in parallel to the original coordinate axis any more. We refer to this data as data set 4. To train the classifiers for the high-dimensional data sets we use the same learning parameter settings as in the previous experiments.

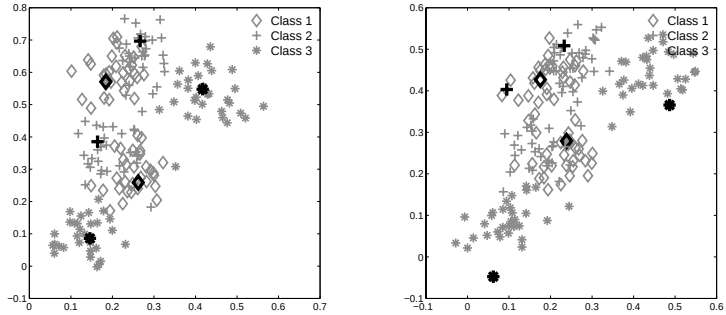
The obtained mean rates of misclassification are reported in Table 2. The results are achieved using the hyperparameter settings $\sigma^2_{opt}(\text{RSLVQ}) = 0.002, 0.003$ and $\sigma^2_{opt}(\text{MRSLVQ}) = 0.003$ for data set 3 and 4, respectively. The performance of GLVQ clearly degrades due to the additional noise in the data. However, by adapting the metric to the structure of the data, GMLVQ is able to achieve nearly the same accuracy on data sets 2 and 3. A visualization of the resulting relevance matrix Λ_{GMLVQ} is provided in Figure 4. The diagonal elements turn out that the algorithm totally eliminates the noisy dimensions 4 to 10, which, in consequence, do not contribute to the computation of distances any more. As reflected by the off-diagonal elements, the classifier additionally takes correlations between the informative dimensions 1 to 3 into account to quantify the similarity of prototypes and feature vectors. Interestingly, the algorithms based on the statistically motivated cost function show strong overfitting effects on this data set. MRSLVQ does not detect the relevant structure in the data sufficiently to reproduce the classification performance achieved on data set 2. The respective relevance matrix trained on data set 3 (see Figure 4) depicts, that the algorithm does not totally prune out the uninformative dimensions. The superiority of GMLVQ in this application is also reflected by the final position of the prototypes in feature space (see Figure 1). A comparable result for GMLVQ can even be observed after training the algorithm on data set 4. Hence, the method succeeds to detect the discriminative structure in the data, even after rotating or scaling the data arbitrarily.



(a) Data set 1. Left: GLVQ prototypes. Right: RSLVQ protoypes

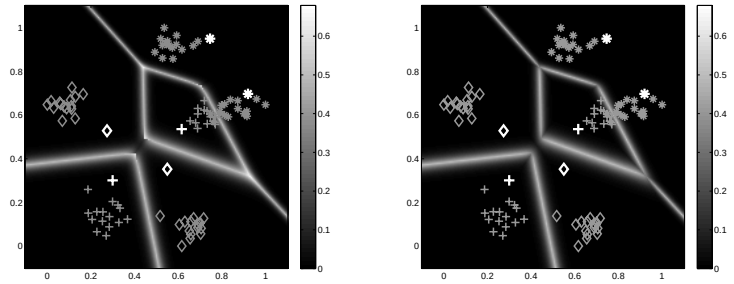


(b) Data set 2. Left: GLVQ prototypes. Right: RSLVQ prototypes

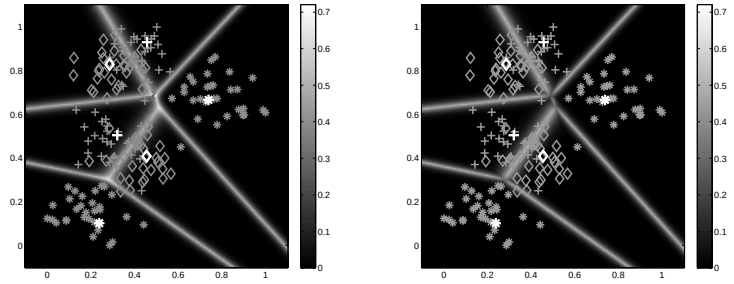


(c) Data set 3. Left: GMLVQ prototypes. Right: MRSLVQ prototypes. The plots relate to the first two dimensions after projecting the data and the prototypes with Ω_{GMLVQ} and Ω_{MRSLVQ}, respectively.

Figure 1: Artificial training data sets and prototype constellations identified by GLVQ, RSLVQ, GMLVQ and MRSLVQ in a single run.



(a) Data set 1. Left: Attractive forces. Right: Repulsive forces. The plots relate to the hyperparameter $\sigma^2 = 0.002$.



(b) Data set 2. Left: Attractive forces. Right: Repulsive forces. The plots relate to the hyperparameter $\sigma^2 = 0.002$.

Figure 2: Visualization of the update factors $(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))$ (attractive forces) and $P(j|\boldsymbol{\xi})$ (repulsive forces) of the nearest prototype with correct and incorrect class label on data sets 1 and 2. It is assumed that every data point is classified correctly.

4.2 Real life data

Image segmentation data set

In a second experiment, we apply the algorithms to the image segmentation data set provided by the UCI repository of Machine Learning [9]. The data set contains 19-dimensional feature vectors, which encode different attributes of 3×3 pixel regions extracted from outdoor images. Each region is assigned to one of seven classes (brickface, sky, foliage, cement, window, path, grass). The features 3-5 are (nearly) constant and are eliminated for these experiments. As a further preprocessing step, the features are normalized to zero mean and unit variance. The provided data is split into a training and a test set (30 samples per class for training, 300 samples per class for testing). In order to find useful values for the hyperparameter in RSLVQ and related methods, we randomly split the test data in a validation and a test set of equal size. The validation set is not used for the experiments with GMLVQ. Each class is approximated by one prototype. We use the parameter settings

(Local) G(M)LVQ: $\alpha_1 = 0.01$, $\alpha_2 = 5 \times 10^{-3}$, $c = 0.001$
(Local) (M)RSLVQ: $\alpha_1 = 0.01$, $\alpha_2 = 1 \times 10^{-3}$, $c = 0.01$

and test values for σ^2 in the interval [0.1, 4.0]. The algorithms are trained for 2000 epochs in total. In the following, we always refer to the experiments with the hyperparameter resulting in the best performance on the validation set. The respective values are $\sigma^2_{opt}(\text{RSLVQ}) = 0.02$, $\sigma^2_{opt}(\text{MRSLVQ}) = 0.75$ and $\sigma^2_{opt}(\text{LMRSLVQ}) = 1.0$.

The obtained classification accuracies are summarized in Table 3. For both cost function schemes the performance improves with increasing complexity of the distance measure, except for Local MRSLVQ which shows overfitting effects. Remarkably, RSLVQ and MRSLVQ clearly outperform the respective GLVQ methods on this data set. Regarding GLVQ and RSLVQ, this observation is solely based on different prototype constellations. The algorithms identify similar $\boldsymbol{w}$ for classes with low rate of misclassification. Differences can be observed in case of prototypes, which contribute strongly to the overall test error. For demonstration purposes, we refer to classes 3 and 7. The mean class specific test errors constitute $\varepsilon^3_{test} = 25.5\%$ and $\varepsilon^7_{test} = 1.2\%$ for the GLVQ classifiers and $\varepsilon^3_{test} = 10.3$ and $\varepsilon^7_{test} = 1.2\%$ for the RSLVQ classifiers. The respective prototypes obtained in one cross validation run are visualized in Figure 3. It depicts that the algorithms identify nearly the same representative for class 7, while the class 3 prototypes reflect differences for the alternative learning strategies. This finding holds similarly for the GMLVQ and MRSLVQ prototypes, however, it is less pronounced (see Figure 3).

The varying classification performance of the two latter methods also goes back to different metric parameter settings derived during training. Comparing the relevance matrices (see Figure 4) shows that GMLVQ and MRSLVQ identify the same dimensions as being most discriminative to classify the data. The features which achieve the highest weight values on the diagonal are the same in both cases. But note, that the feature selection by MRSLVQ is more pronounced. Interestingly, differences in the

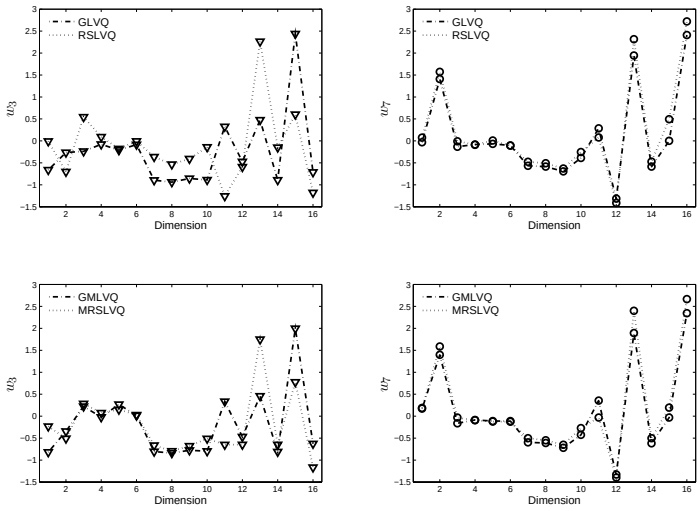


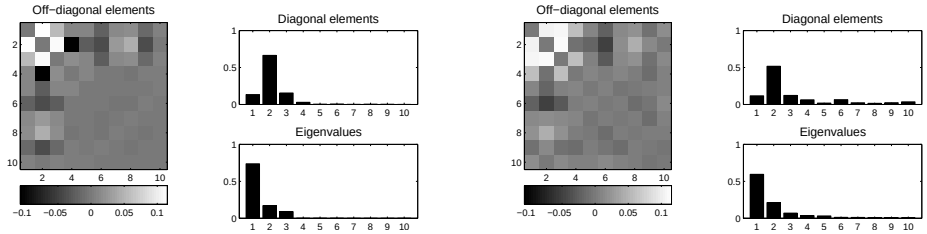
Figure 3: Visualization of the class 3 and class 7 prototypes of the image segmentation data set. Top: Prototypes identified by GLVQ and RSLVQ. Buttom: Prototypes identified by GMLVQ and MRSLVQ.

prototype configurations mainly occur in the dimensions evaluated as most important for classification.

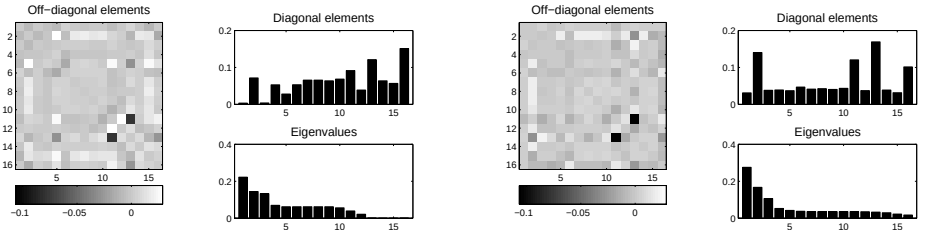
Finally, we discuss how the performance of RSLVQ, MRSLVQ and Local MRSLVQ depends on the value of the hyperparameter. Figure 5 displays the evolution of the mean final validation errors with varying σ^2. It can be observed that the value σ^2_{opt}, where the curves reach their minimum, increases with the complexity of the distance measure. Furthermore, the range of σ^2 achieving an accuracy close to the performance of σ^2_{opt} becomes wider for MRSLVQ and Local MRSLVQ, while the RSLVQ curve shows a very sharp minimum. Hence, it can be stated that the methods become less sensitive with respect to the hyperparameter, if an advanced metric is used to quantify the similarity between prototypes and feature vectors. For σ^2 close to zero, all algorithms show instabilities and highly fluctuating learning curves.

Letter data set

The Letter data set from the UCI repository [9] consists of 20 000 feature vectors which encode 16 numerical attributes of black-and-white rectangular pixel displays of the 26 capital letters of the English alphabet. The features are scaled to fit into a range of integer values between 0 and 15. This data set is also used in [12] to analyse the performance of RSLVQ. We extract one half of the samples of each class for training the classifiers and one fourth for testing and validating, respectively. The following results are averaged over 10 independent constellations of the different data sets. We train



(a) Data set 3. Left: GMLVQ matrix. Right: MRSLVQ matrix



(b) Image segmentation data set. Left: GMLVQ matrix. Right: MRSLVQ matrix

Figure 4: Visualization of the relevance matrices Λ obtained during GMLVQ- and MRSLVQ-training when applied to the artificial data set 3 and the image segmentation data set in a single run. The elements Λ_{ii} are set to zero in the visualization of the off-diagonal elements. The matrices in 4(b) are normalized to $\sum_i \Lambda_{ii} = 1$ after training.

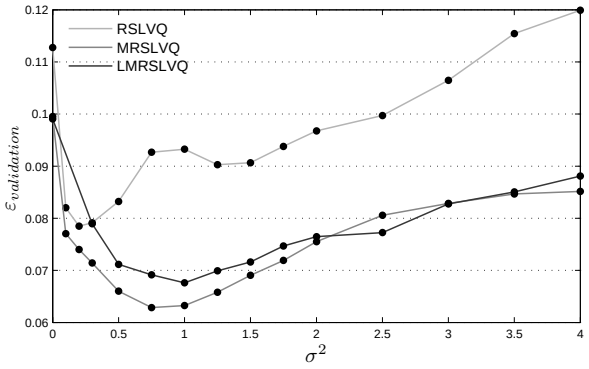


Figure 5: Mean validation errors obtained on the image segmentation data set by RSLVQ, MRSLVQ and Local MRSLVQ using different setting of the hyperparameters σ^2.

the classifiers with one prototype per class respectively and use the learning parameter settings

(Local) G(M)LVQ: $\alpha_1 = 0.1$, $\alpha_2 = 0.01$, $c = 0.1$

(Local) (M)RSLVQ: $\alpha_1 = 0.01$, $\alpha_2 = 0.001$, $c = 0.1$.

Training is continued for 150 epochs in total with different values σ^2 lying in the interval [0.75, 4.0]. The accuracy on the validation set is used to select the best settings for the hyperparameter. With the settings $\sigma^2_{opt}(\text{RSLVQ}) = 1.0$ and $\sigma^2_{opt}(\text{MRSLVQ}, \text{Local MRSLVQ}) = 1.5$ we achieve the performances stated in Table 3. The results depict that training of an individual metric for every prototype is particularly efficient in case of multi-class problems. The adaptation of a global relevance matrix does not provide significant benefit because of the huge variety of classes in this application. Similar to the previous application, the RSLVQ-based algorithms outperform the methods based on the GLVQ cost function. Remarkably, the classification accuracy of Local MRSLVQ with one prototype per class is comparable to the RSLVQ results presented in [12], achieved with constant hyperparameter σ^2 and 13 prototypes per class. This observation underlines the crucial importance of an appropriate distance measure for the performance of LVQ-classifiers. Despite the large number of parameters, we do not observe overfitting effects during training of local relevance matrices on this data set. The systems show stable behaviour and converge within 100 training epochs.

5 Conclusions

We have considered metric learning by matrix adaptation in discriminative vector quantization schemes. In particular, we have introduced this principle into soft robust learning vector quantization, which is based on an explicit statistical model by means of mixtures of Gaussians, and we extensively compared this method to an alternative scheme derived from an intuitive but somewhat heuristic cost function. In general, it can be observed that matrix adaptation allows to improve the classification accuracy on the one hand, and it leads to a simplification of the classifier and thus better interpretability of the results by inspection of the eigenvectors and eigenvalues on the other hand. Interestingly, the behavior of GMLVQ and MRSLVQ shows several principled differences. Based on the experimental findings, the following conclusions can be drawn:

- All discriminative vector quantization schemes show good generalization behavior and yield reasonable classification accuracy on several benchmark results using only few prototypes. RSLVQ seems particularly suited for the real-life data sets considered in this article. In general, matrix learning allows to further improve the results, whereby, depending on the setting, overfitting can be more pronounced due to the huge number of free parameters.
- The methods are generally robust against noise in the data as can be inferred from different runs of the algorithm on different splits of the data sets. While

GLVQ and variants are rather robust to the choice of hyperparameters, a very critical hyperparameter of training is the softness parameter σ^2 for RSLVQ. Matrix adaptation seems to weaken the sensitivity w.r.t. this parameter, however, a correct choice of σ^2 is still crucial for the classification accuracy and efficiency of the runs. For this reason, automatic adaptation schemes for σ^2 should be considered. In [13], a simple annealing scheme for σ^2 is introduced which yields reasonalbe results. We are currently working on a scheme which adapts σ^2 in a more principled way according to an optimization of the likelihood ratio showing first promising results.

- The methods allow for an inspection of the classifier by means of the prototypes which are defined in input space. Note that one explicit goal of *unsupervised* vector quantization schemes such as k-means or the self-organizing map is to represent typical data regions be means of prototypes. Since the considered approaches are discriminative, it is not clear in how far this property is maintained for GLVQ and RSLVQ variants. The experimental findings demonstrate that GLVQ schemes place prototypes close to class centres and prototypes can be interpreted as typical class representatives. On the contrary, RSLVQ schemes do not preserve this property in particular for non-overlapping classes since adaptation basically takes place based on misclassifications of the data. Therefore, prototypes can be located outside the class centers while maintaining the same or a similar classification boundary compared to GLVQ schemes. This property has already been observed and proven in typical model situations using the theory of online learning for the limit learning rule of RSLVQ, learning from mistakes, in [2].

- Despite the fact that matrix learning introduces a huge number of additional free parameters, the method tends to yield very simple solutions which involve only few relevant eigendirections. This behavior can be substantiated by an exact mathematical investigation of the LVQ2.1-type limit learning rules which result for small σ^2 or a steep sigmoidal function Φ, respectively. For these limits, an exact mathematical investigation becomes possible, indicating that a unique solution for matrix learning exist, given fixed prototypes, and that the limit matrix reduces to a singular matrix which emphasizes one major eigenvalue direction. The exact mathematical treatment of these simplified limit rules is subject of ongoing work and will be published in subsequent work.

In conclusion, systematic differences of GLVQ and RSLVQ schemes result from the different cost functions used in the approaches. This includes a larger sensitivity of RSLVQ to hyperparanmeters, a different location of prototypes which can be far from the class centres for RSLVQ, and different classification accuracies in some cases. Apart from these differences, matrix learning is clearly beneficial for both discriminative vector quantization schemes as demonstrated in the experiments.

A Derivatives

We compute the derivatives of the RSLVQ cost function with respect to the prototypes, the metric parameters, and the hyperparameters. More generally, we compute the derivative of the likelihood ratio with respect to any parameter $\Theta_i \neq \boldsymbol{\xi}$. The conditional densities can be chosen to have the normalized exponential form $p(\boldsymbol{\xi}|j) = K(j) \cdot \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)$. Note that the normalization factor $K(j)$ depends on the shape of component j. If a mixture of N-dimensional Gaussian distributions is assumed, $K(j) = (2\pi\sigma_j^2)^{(-N/2)}$ is only valid under the constraint $\det(\Lambda_j) = 1$. We point out that the following derivatives subject to the condition $\det(\Lambda_j) = const.\ \forall j$. With $\det(\Lambda_j) = const.\ \forall j$, the $K(j)$ as defined above are scaled by a constant factor which can be disregarded. The condition of equal determinant for all j naturally includes the adaptation of a global relevance matrix $\Lambda = \Lambda_j, \forall j$.

$$
\begin{aligned}
&\frac{\partial}{\partial \Theta_i}\left(\log \frac{p(\boldsymbol{\xi}, y|W)}{p(\boldsymbol{\xi}|W)}\right) \\
&= \frac{p(\boldsymbol{\xi}|W)}{p(\boldsymbol{\xi}, y|W)}\left(\frac{1}{p(\boldsymbol{\xi}|W)}\frac{\partial p(\boldsymbol{\xi}, y|W)}{\partial \Theta_i} - \frac{p(\boldsymbol{\xi}, y|W)}{p(\boldsymbol{\xi}|W)^2}\frac{\partial p(\boldsymbol{\xi}|W)}{\partial \Theta_i}\right) \\
&= \frac{1}{p(\boldsymbol{\xi}, y|W)}\underbrace{\frac{\partial p(\boldsymbol{\xi}, y|W)}{\partial \Theta_i}}_{(a)} - \frac{1}{p(\boldsymbol{\xi}|W)}\left(\underbrace{\frac{\partial p(\boldsymbol{\xi}, y|W)}{\partial \Theta_i}}_{(a)} + \underbrace{\textstyle\sum_{c\neq y}\frac{\partial p(\boldsymbol{\xi}, c|W)}{\partial \Theta_i}}_{(b)}\right) \\
&= \delta_{y,c(\boldsymbol{w}_i)}\left(\frac{P(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}, y|W)}\frac{\partial K(i)}{\partial \Theta_i}\right. \\
&\qquad\left. + \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}, y|W)}\ \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i}\right) \\
&- \delta_{y,c(\boldsymbol{w}_i)}\left(\frac{P(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|W)}\frac{\partial K(i)}{\partial \Theta_i}\right. \\
&\qquad\left. + \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|W)}\ \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i}\right) \\
&- (1 - \delta_{y,c(\boldsymbol{w}_i)})\left(\frac{P(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|W)}\frac{\partial K(i)}{\partial \Theta_i}\right. \\
&\qquad\left. + \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|W)}\ \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i}\right)
\end{aligned}
\tag{24}
$$

$$= \delta_{y,c(\boldsymbol{w}_i)} \left(P_y(i|\boldsymbol{\xi}) - P(i|\boldsymbol{\xi})\right) \left(\frac{1}{K(i)} \frac{\partial K(i)}{\partial \Theta_i} + \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i}\right)$$
$$- \left(1 - \delta_{y,c(\boldsymbol{w}_i)}\right) P(i|\boldsymbol{\xi}) \left(\frac{1}{K(i)} \frac{\partial K(i)}{\partial \Theta_i} + \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i}\right) \tag{25}$$

with (a)

$$\begin{aligned}
&\frac{\partial p(\boldsymbol{\xi}, y|W)}{\partial \Theta_i} \\
&= \frac{\partial}{\partial \Theta_i} \Big(\sum_{j:c(\boldsymbol{w}_j)=y} P(j)\, p(\boldsymbol{\xi}|j) \Big) \\
&= \sum_j \delta_{y,c(\boldsymbol{w}_j)}\, P(j)\, \frac{\partial p(\boldsymbol{\xi}|j)}{\partial \Theta_i} \\
&= \sum_j \delta_{y,c(\boldsymbol{w}_j)}\, P(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j) \left(\frac{\partial K(j)}{\partial \Theta_i} + K(j) \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)}{\partial \Theta_i}\right)
\end{aligned}$$

and (b)

$$\begin{aligned}
&\sum_{c \neq y} \frac{\partial p(\boldsymbol{\xi}, c|W)}{\partial \Theta_i} \\
&= \frac{\partial}{\partial \Theta_i} \Big(\sum_{j:c(\boldsymbol{w}_j)\neq y} P(j)\, p(\boldsymbol{\xi}|j) \Big) \\
&= \sum_j (1 - \delta_{y,c(\boldsymbol{w}_j)})\, P(j)\, \frac{\partial p(\boldsymbol{\xi}|j)}{\partial \Theta_i} \\
&= \sum_j (1 - \delta_{y,c(\boldsymbol{w}_j)})\, P(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j) \left(\frac{\partial K(j)}{\partial \Theta_i} + K(j) \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)}{\partial \Theta_i}\right)
\end{aligned}$$

$P_y(i|\boldsymbol{\xi})$ and $P(i|\boldsymbol{\xi})$ are assignment probabilities,

$$\begin{aligned} P_y(i|\boldsymbol{\xi}) &= \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}, y|W)} \\ &= \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\sum_{j:c(\boldsymbol{w}_j)=y} P(j)K(j)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)} \end{aligned}$$

$$\begin{aligned} P(i|\boldsymbol{\xi}) &= \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|W)} \\ &= \frac{P(i)K(i)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\sum_j P(j)K(j)\exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)} \end{aligned}$$

$P_y(i|\boldsymbol{\xi})$ constitutes the probability that sample $\boldsymbol{\xi}$ is assigned to component i of the correct class y and $P(i|\boldsymbol{\xi})$ depicts the probability the $\boldsymbol{\xi}$ is assigned to any component i of the mixture.
The derivative with respect to a global parameter, e.g. a global matrix $\Omega = \Omega_j$ for all j can be derived thereof by summation.

References

[1] B. Arnonkijpanich, B. Hammer, A. Hasenfuss, and A. Lursinap. Matrix learning for topographic neural maps. In *ICANN'2008*, 2008.

[2] M. Biehl, A. Ghosh, and B. Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, 8:323–360, 2007.

[3] T. Bojer, B. Hammer, D. Schunk, and K. Tluk von Toschanowitz. Relevance determination in learning vector quantization. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks*, pages 271–276, 2001.

[4] K. Crammer, R. Gilad-Bachrach, A. Navot, and A. Tishby. Margin analysis of the lvq algorithm. In *Advances in Neural Information Processing Systems*, volume 15, pages 462–469. MIT Press, Cambridge, MA, 2003.

[5] B. Hammer, M. Strickert, and T. Villmann. On the generalization ability of GRLVQ networks. *Neural Processing Letters*, 21(2):109–120, 2005.

[6] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[7] Samuel Kaski. Principle of learning metrics for exploratory data analysis. In *Neural Networks for Signal Processing XI, Proceedings of the 2001 IEEE Signal Processing Society Workshop*, pages 53–62. IEEE, 2001.

[8] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, second edition, 1997.

[9] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. Uci repository of machine learning databases. `http://archive.ics.uci.edu/ml/`, 1998.

[10] A. Sato and K. Yamada. Generalized learning vector quantization. In M. C. Mozer D. S. Touretzky and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–9, Cambridge, MA, USA, 1996. MIT Press.

[11] P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. Submitted, 2007.

[12] Sambu Seo and Klaus Obermayer. Soft learning vector quantization. *Neural Computation*, 15(7):1589–1604, 2003.

[13] Sambu Seo and Klaus Obermayer. Dynamic hyper parameter scaling method for lvq algorithms. In *International Joint Conference on Neural Networks*, Vancouver, Canada, 2006.

[14] M. Strickert, K. Witzel, H.-P. Mock, F.-M. Schleif, and T. Villmann. Supervised attribute relevance determination for protein identification in stress experiments. In *Proceedings of Machine Learning in Systems Biology*, 2007.

[15] Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA, 2006.

Table 2: Mean rate of misclassification (in %) obtained by the different algorithms on the artificial data sets 3 and 4 at the end of training. The values in brackets constitute the variances.

Algorithm	Data set 3		Data set 4	
	ε_{train}	ε_{test}	ε_{train}	ε_{test}
GLVQ	23.5 (0.1)	38.0 (0.2)	31.2 (0.06)	41.0 (0.2)
GMLVQ	12.1 (0.1)	24.0 (0.36)	14.5 (0.1)	30.6 (0.5)
RSLVQ	4.1 (0.1)	33.2 (0.5)	11.7 (0.1)	36.8 (0.17)
MRSLVQ	3.9 (0.08)	29.5 (0.4)	8.0 (0.08)	32.0 (0.22)

Table 3: Mean rate of misclassification (in %) obtained by the different algorithms on the image segmentation and letter data set at the end of training. The values in brackets constitute the variances.

Algorithm	Image segmentation data		Letter data	
	ε_{train}	ε_{test}	ε_{train}	ε_{test}
GLVQ	15.2 (0.0)	17.0 (0.003)	28.4 (10^{-3})	28.9 (10^{-4})
GMLVQ	9.1 (0.002)	10.2 (0.004)	28.3 (10^{-4})	28.8 (0.003)
LGMLVQ	4.8 $(2 \cdot 10^{-4})$	8.6 (0.004)	15.2.4 (10^{-3})	16.9 (0.007)
RSLVQ	1.4 (0.003)	7.5 (0.003)	21.9 (10^{-3})	23.2 (0.005)
MRSLVQ	1.8 (0.002)	6.2 (0.002)	21.7 (10^{-4})	22.8 (0.002)
LMRSLVQ	1.9 (0.0)	6.8 (0.004)	1.3 (10^{-4})	6.2 (10^{-3})